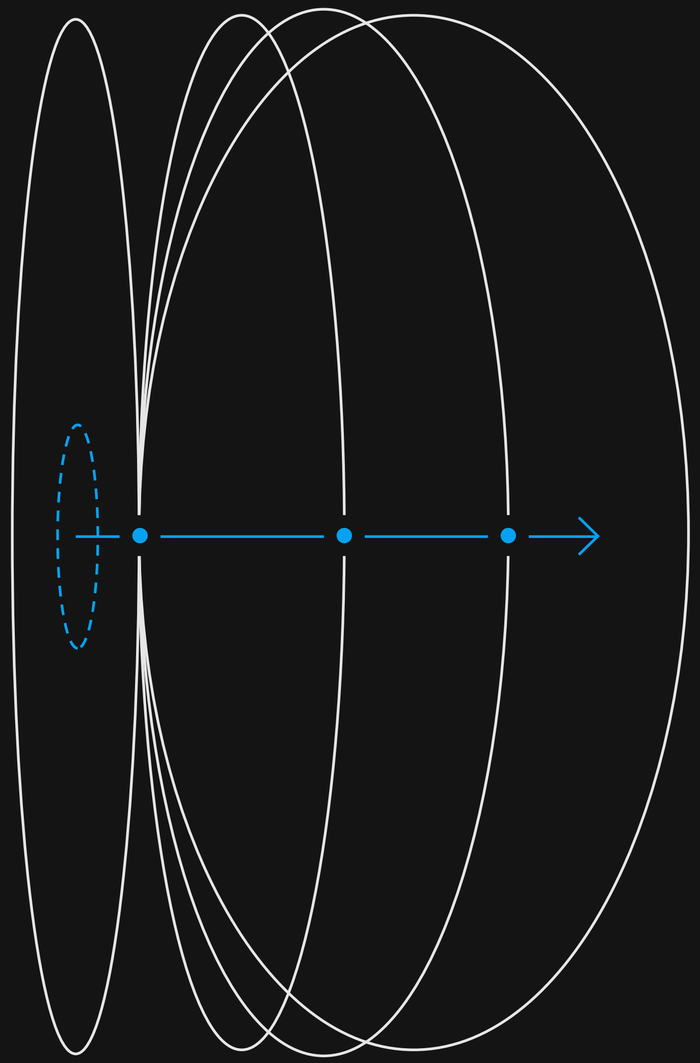
ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

LLMs Operations (LLMOps)

Building ML Solutions

Session 8

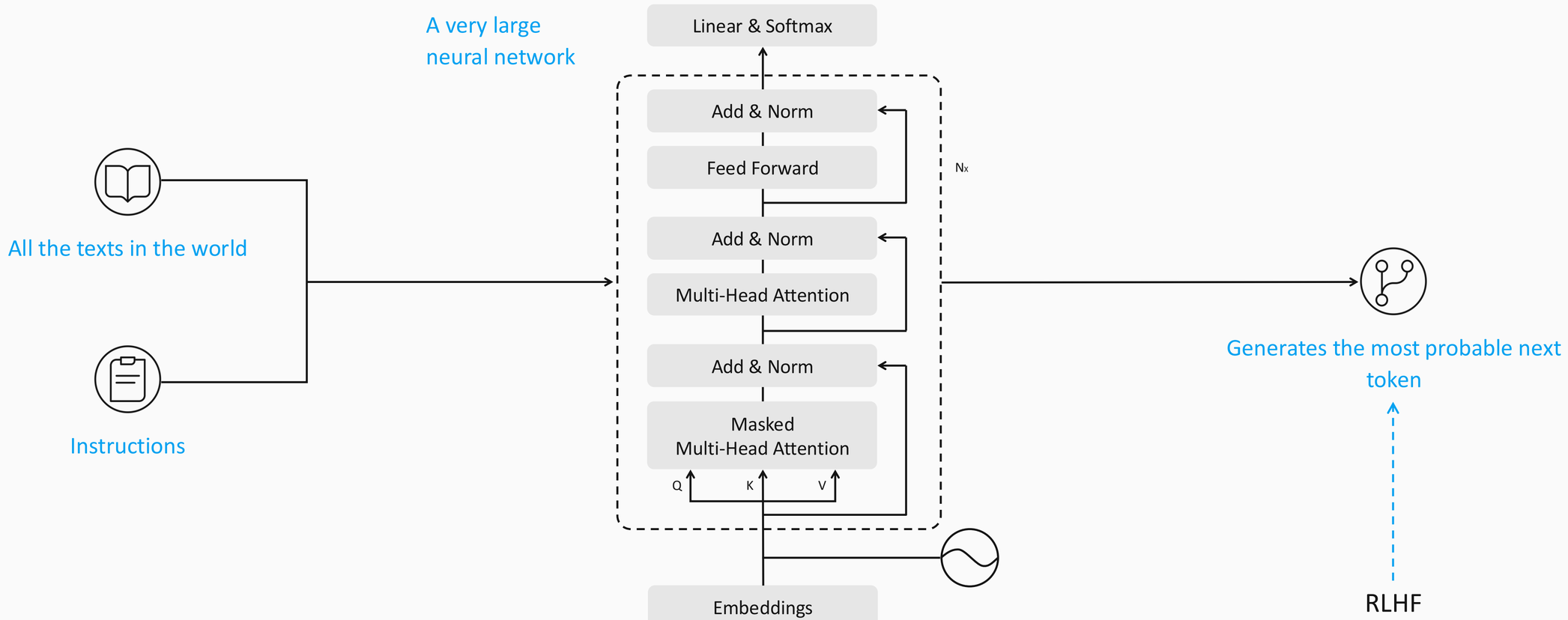
Lecture plan



- What is an LLM?
- Open- vs closed-source LLMs, use cases, APIs
- Infrastructure for local deployment, tools, practice
- Resource savings, quantization, practice
- Configuration specifics, usage nuances, monitoring

What is an LLM?

How it is built



State of GPT — Andrej Karpathy

OpenAI API

```
import openai
from httpx import Client

HTTP_PROXY = f"http://{user}:{password}@{host}:{port}”
API_KEY = "some_api_key"
MODEL_NAME = "gpt4o"

client = openai.OpenAI(api_key=API_KEY, http_client=Client(proxies=HTTP_PROXY))
prompt = " What is CUDA? Return the answer in JSON format "

chat = client.chat.completions.create(model=MODEL_NAME,
                                      messages=[{"role": "user", "content": prompt}])

print(chat.choices[0].message.content)
```

Not only OpenAI: Commercial vs Open Source

- ChatGPT — Trendsetters
- ANTHROPIC CLAUDE 3 — Teams driving cutting-edge research
- Gemini Google — High quality models
- deepseek
- MISTRAL AI_ — Revolutionaries
- Qwen — Awesome open-source
- Meta Llama 3

Why do we need all of this?

Why don’t we just pay for OpenAI/Google

Sensitive data

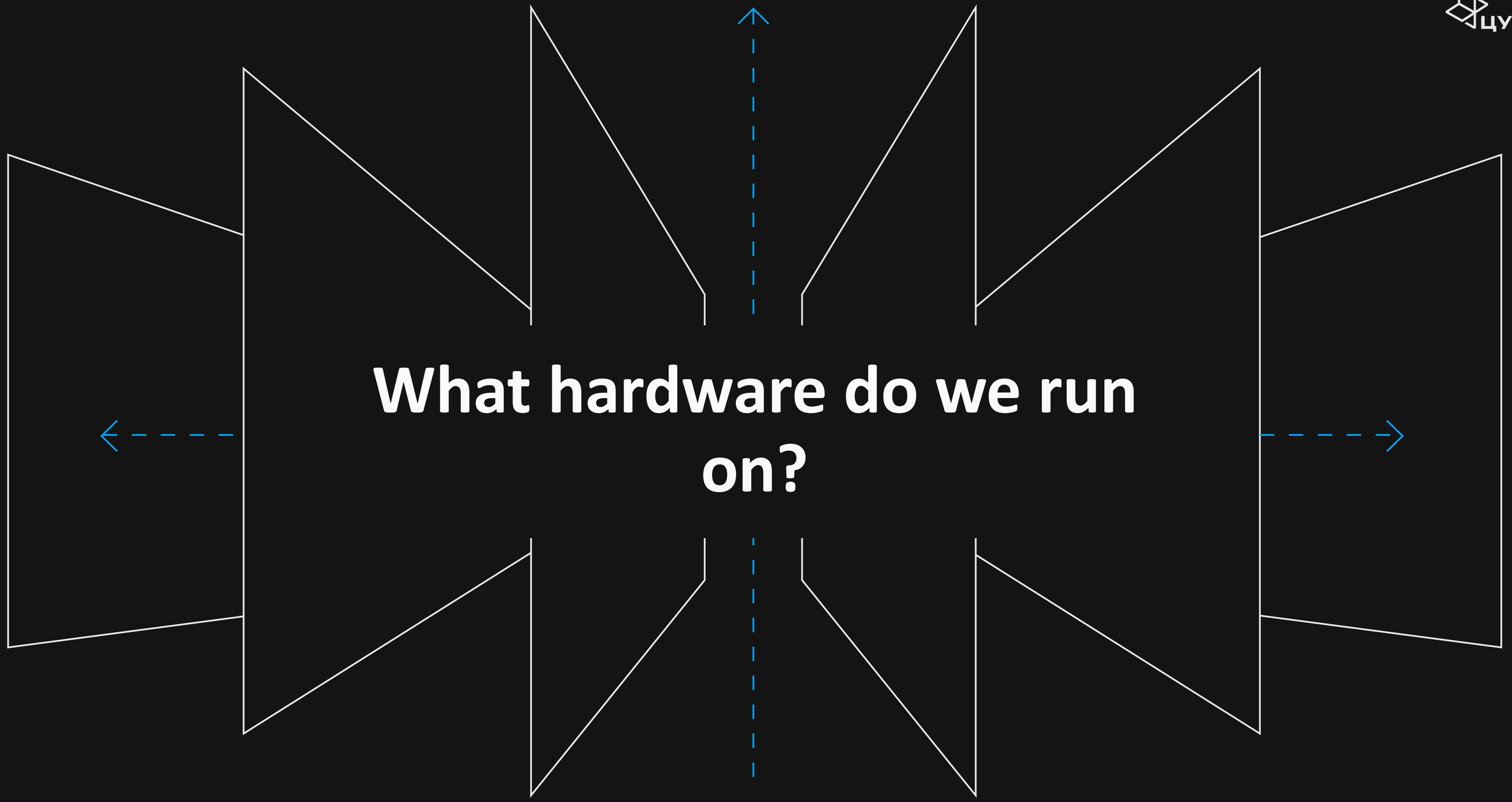
Enterprises avoid sending sensitive data to to third parties clouds like OpenAI

Fine-tuning

Sometimes you need a domain-specific model; you can continue training on your own data.

Cost savings

Self-hosting lets you balance speed and cost as you wish



What hardware do we run on?

System basics

Any Linux distro (Ubuntu for example)

System basics

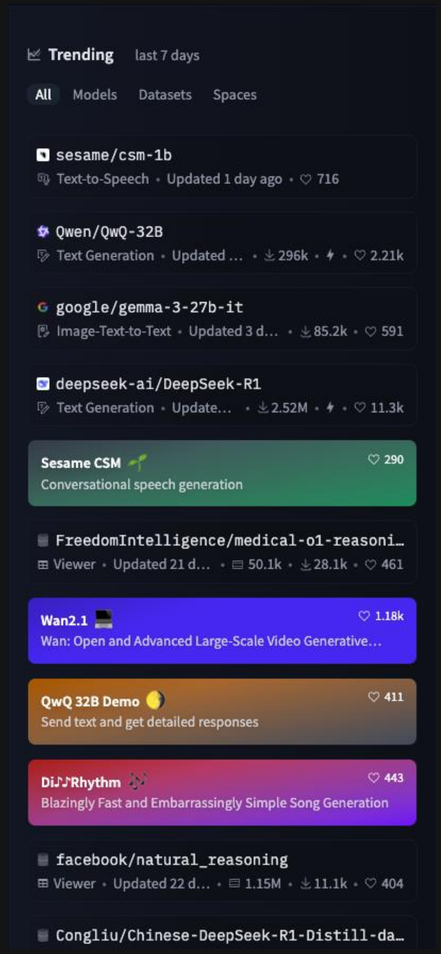
Any Linux distro (Ubuntu for example)

NVIDIA GPU

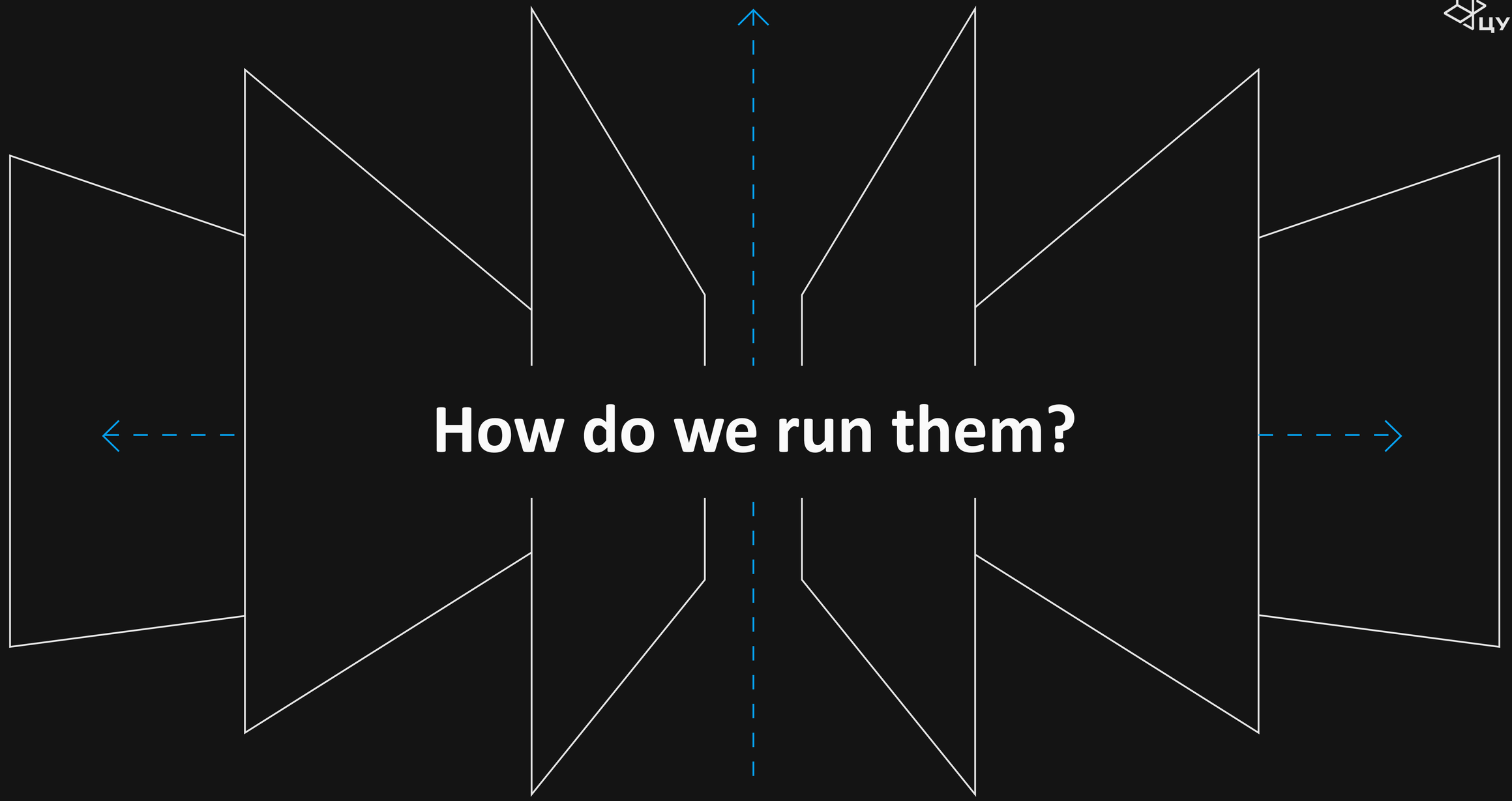
NVIDIA CUDA (CONTAINER) TOOLKIT

Where can we get open models?

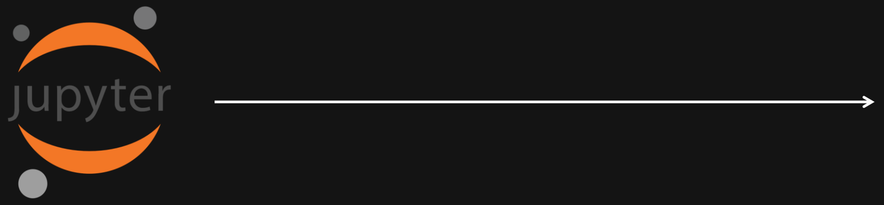
Where can we get open models?



https://huggingface.co/docs/hub/en/models-the-hub


How do we run them?

How do we run them?




```
from useful_framework import something_to_serve_LLM

LLM = something_to_serve_LLM(model_name_or_path)
LLM.generate(query)
```

How do we run them?


```
from useful_framework import something_to_serve_LLM

LLM = something_to_serve_LLM(model_name_or_path)
LLM.generate(query)
```

```
useful_framework serve ./llm_files
python -m useful_framework ./llm_files
```

How do we run them?


```
from useful_framework import something_to_serve_LLM

LLM = something_to_serve_LLM(model_name_or_path)
LLM.generate(query)
```

```
useful_framework serve ./llm_files
python -m useful_framework ./llm_files
```


```
sudo docker run -d…

sudo docker compose up -d
```

Tools for running an LLM

vLLM

Advantages

- Very fast and efficient inference
- Extensive documentation
- Large and active community
- OpenAI-style API (de-facto standard)

Note

- Not every model is supported
- Requires some manual tuning
- Uses all available VRAM by default

Documentation

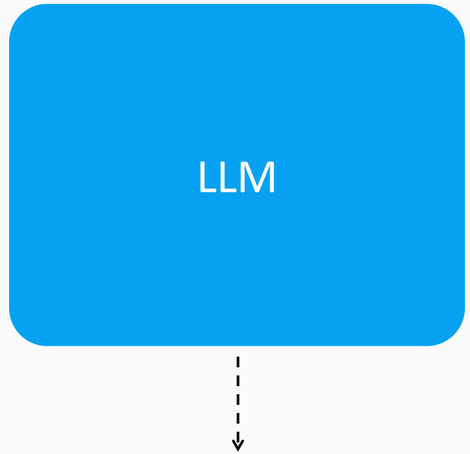
- https://github.com/vllm-project/vllm
- https://docs.vllm.ai/en/latest/

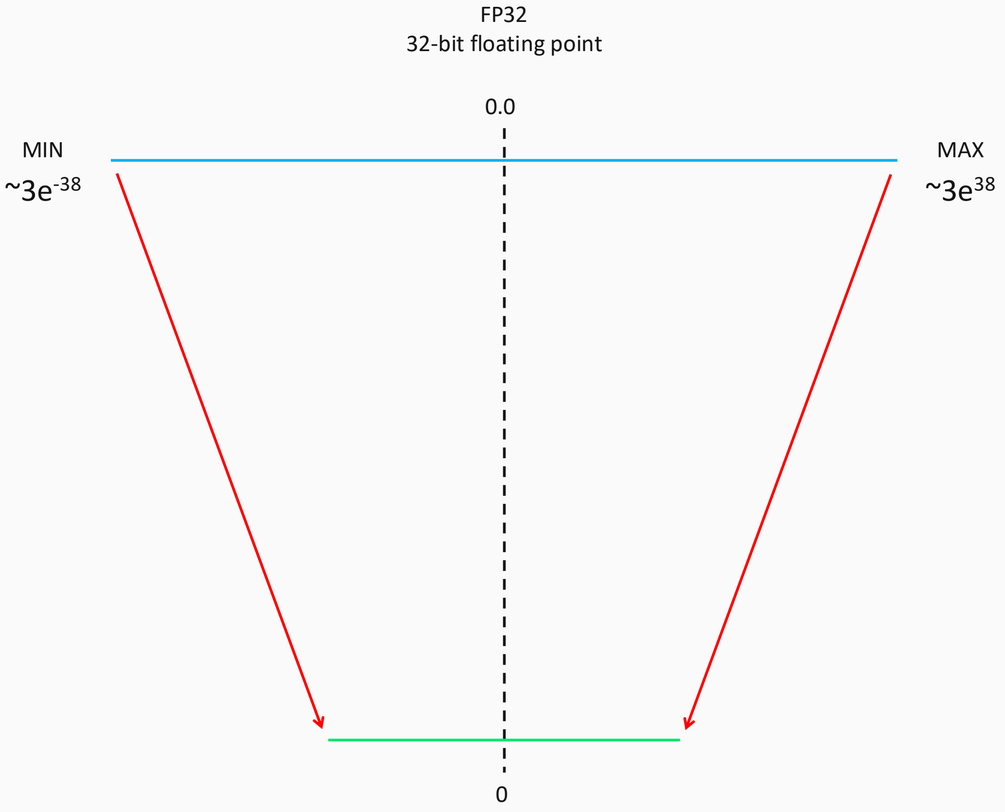
```
#python/jupyter
from vllm import LLM, SamplingParams

#cli
vllm serve Qwen/Qwen2.5-1.5B-Instruct

#python module
python3 -m vllm.serve…

# Docker (preferred & convenient)
docker run --runtime nvidia --gpus all -v
~/.cache/huggingface:/root/.cache/huggingface --env
"HUGGING_FACE_HUB_TOKEN=<secret>" -p 8000:8000 --ipc=host
vllm/vllm-openai:latest --model Qwen/Qwen2.5-1.5B-Instruct

# Calling through openai-compatible endpoint
import openai

BASE_URL = f"http://{host}:{port}"
API_KEY = ”llama_cpp_key"
MODEL_NAME = ”your_model_name"
client = openai.OpenAI(api_key=API_KEY, base_url=BASE_URL)
```

Practice

Quantisation & llama.cpp

some_LLM_modelname.gguf
Georgy Gerganov Unified Format





Используется в учебных целях.
llama.cpp specific quantization types (_K, _M etc.): https://github.com/ggml-org/llama.cpp/pull/1684
Концепция в целом (gguf далеко не единственный вариант): https://huggingface.co/docs/optimum/en/concept_guides/quantization

llama.cpp

Advantages

- Cost-effective; you can host several models at once
- Multiple quantisation levels to choose from
- Very convenient: 1 file = 1 model
- Runs on both CPU **and** GPU

Note

- Lower quality; may freeze under load
- The author updates the framework almost daily, so images change frequently
- Sits under the hood of many LLM apps (Ollama, LM Studio, etc.);

Documentation

- https://github.com/ggerganov/llama.cpp
- https://github.com/ggml-org/llama.cpp/blob/master/tools/server/README.md

```
docker run -p 8080:8080 -v /path/to/models:/models --gpus all
ghcr.io/ggerganov/llama.cpp:server-cuda -m models/7B/ggml-model.gguf -c 512
--host 0.0.0.0 --port 8080 --n-gpu-layers 99 --api-key llama_cpp_key


curl --request POST \ --url http://localhost:8080/completion \ --header
"Content-Type: application/json" \ --data '{"prompt": "Building a website
can be done in 10 simple steps:", "n_predict": 128}'


import openai

BASE_URL = f"http://{host}:{port}"
API_KEY = "llama_cpp_key"
MODEL_NAME = "your_model_name"
client = openai.OpenAI(api_key=API_KEY,
                       base_url=BASE_URL)  #may AsyncOpenAI

prompt = "What is CUDA? Return the answer in JSON format"
chat = client.chat.completions.create(model=MODEL_NAME,
                                      messages=[{"role": "user", "content":
                                                 prompt}])

print(chat.choices[0].message.content)
```

Practice

Infinity Embeddings

Advantages

- Extremely fast and efficient inference
- GPU support
- Deploy many models simultaneously
- OpenAI-like embeddings API
- Includes /rerank and /classify endpoints

Note

- Some models may be unsupported

Documentation

- https://infinity.modal.michaelfeil.eu
- https://github.com/michaelfeil/infinity

```
docker run -d --gpus all -v /path/to/models:/app/.cache -p 8005:8005
michaelf34/infinity:latest v2 --model-id /app/.cache/bge-m3 --model-id
/app/.cache/bge-reranker-v2-m3 --port 8005 --api-key embeddings_key
```

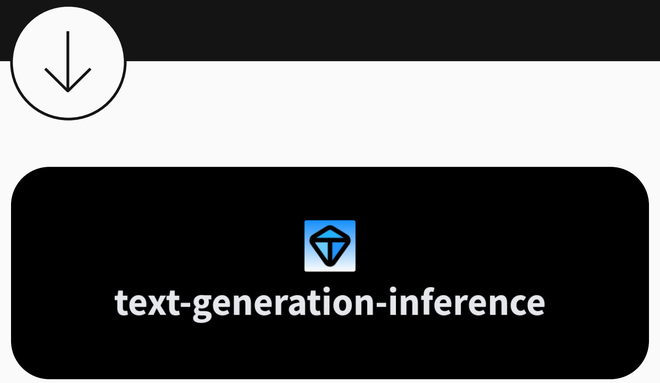
```
import openai

BASE_URL = f"http://{host}:{port}"
API_KEY = "embeddings_key"
client_emb = openai.OpenAI(api_key=API_KEY,
                           base_url=BASE_URL) #may AsyncOpenAI

responses = client_emb.embeddings.create(input=["Hello",
                                                "Hi"],
                                         model=model)
for data in responses.data:
        print(data.embedding[:5])
```

* https://huggingface.co/spaces/mteb/leaderboard

Practice

Haven’t we invented anything else?

https://github.com/aphrodite-engine/aphrodite-engine

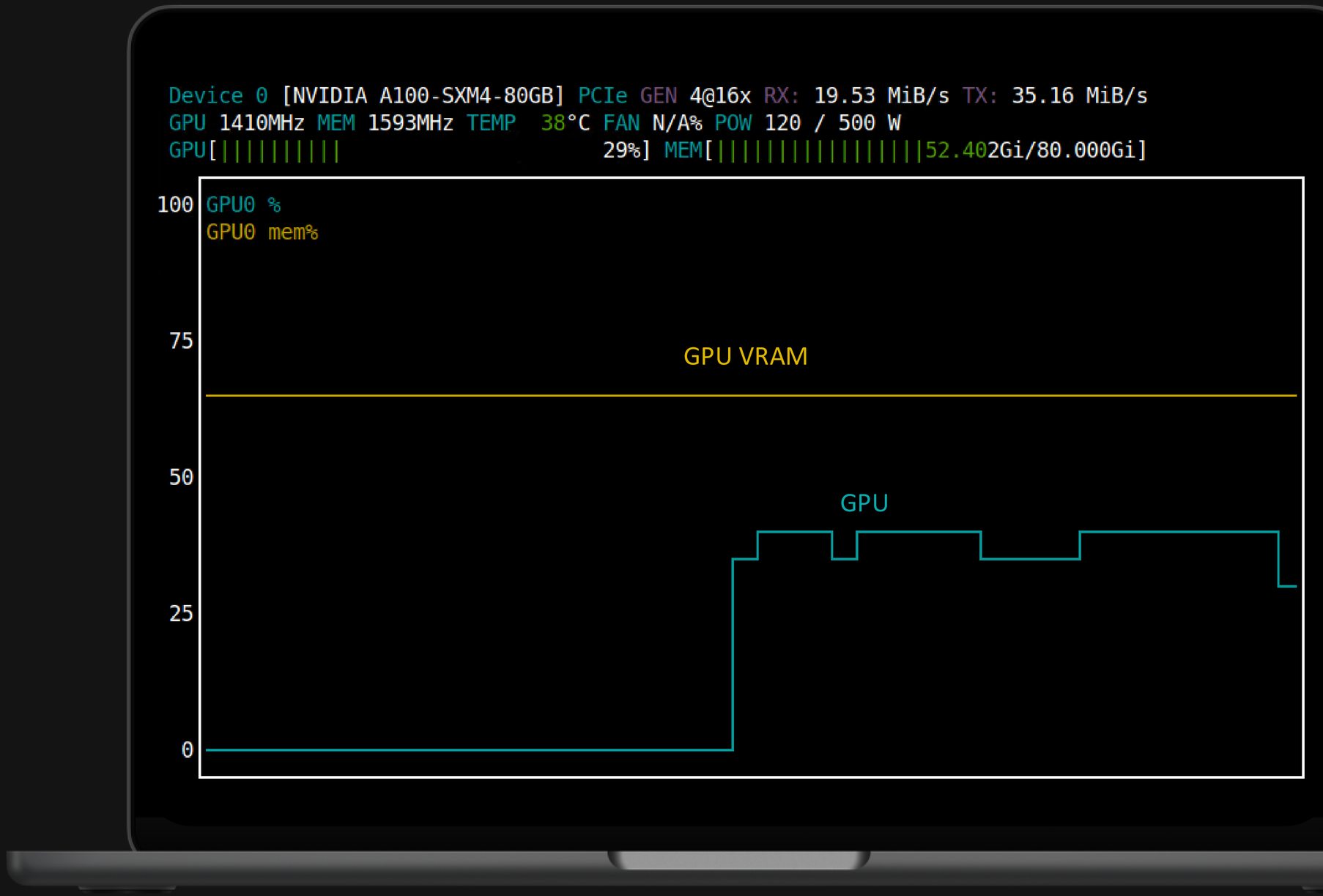


https://github.com/huggingface/text-generation-inference

https://github.com/sgl-project/sglang

Resource monitoring

- nvtop
- Grafana



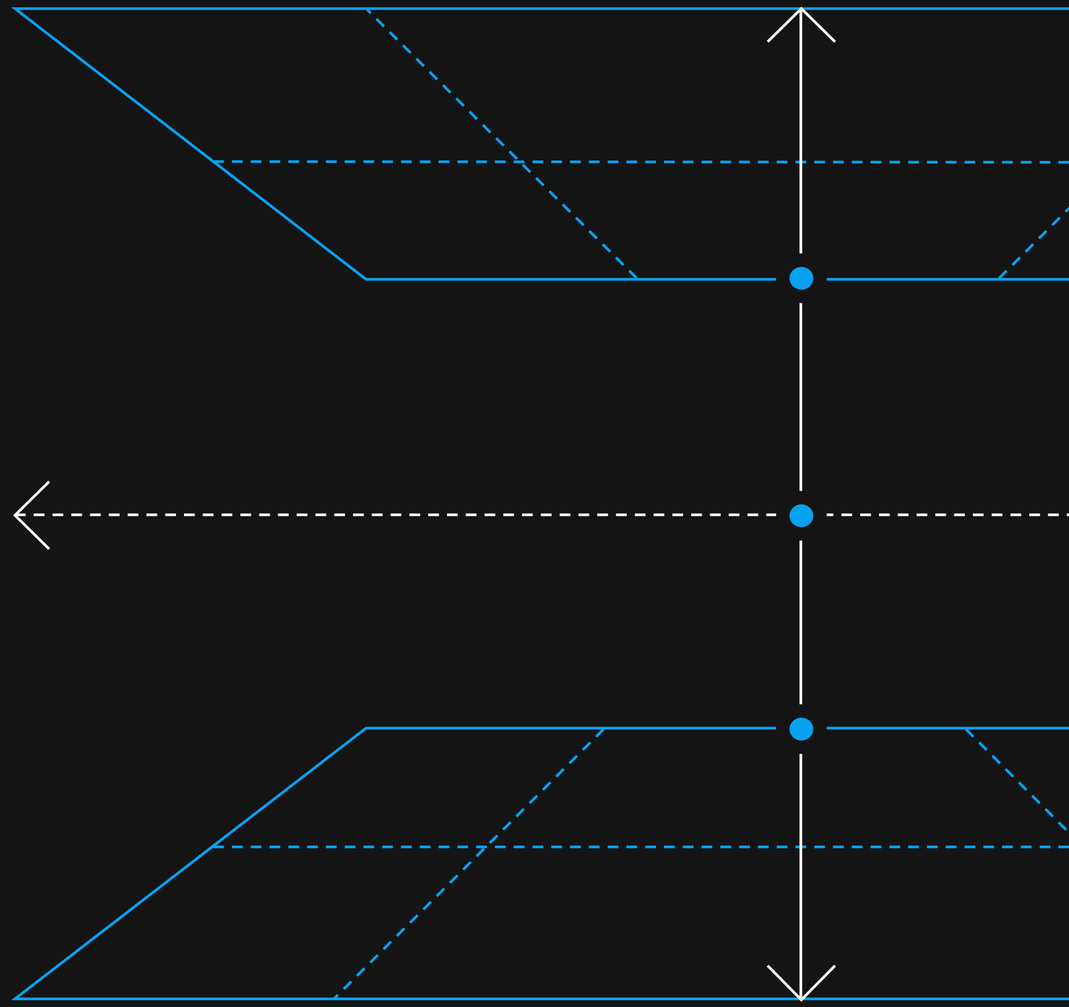
How to estimate resources beforehand

To run in full byte precision:

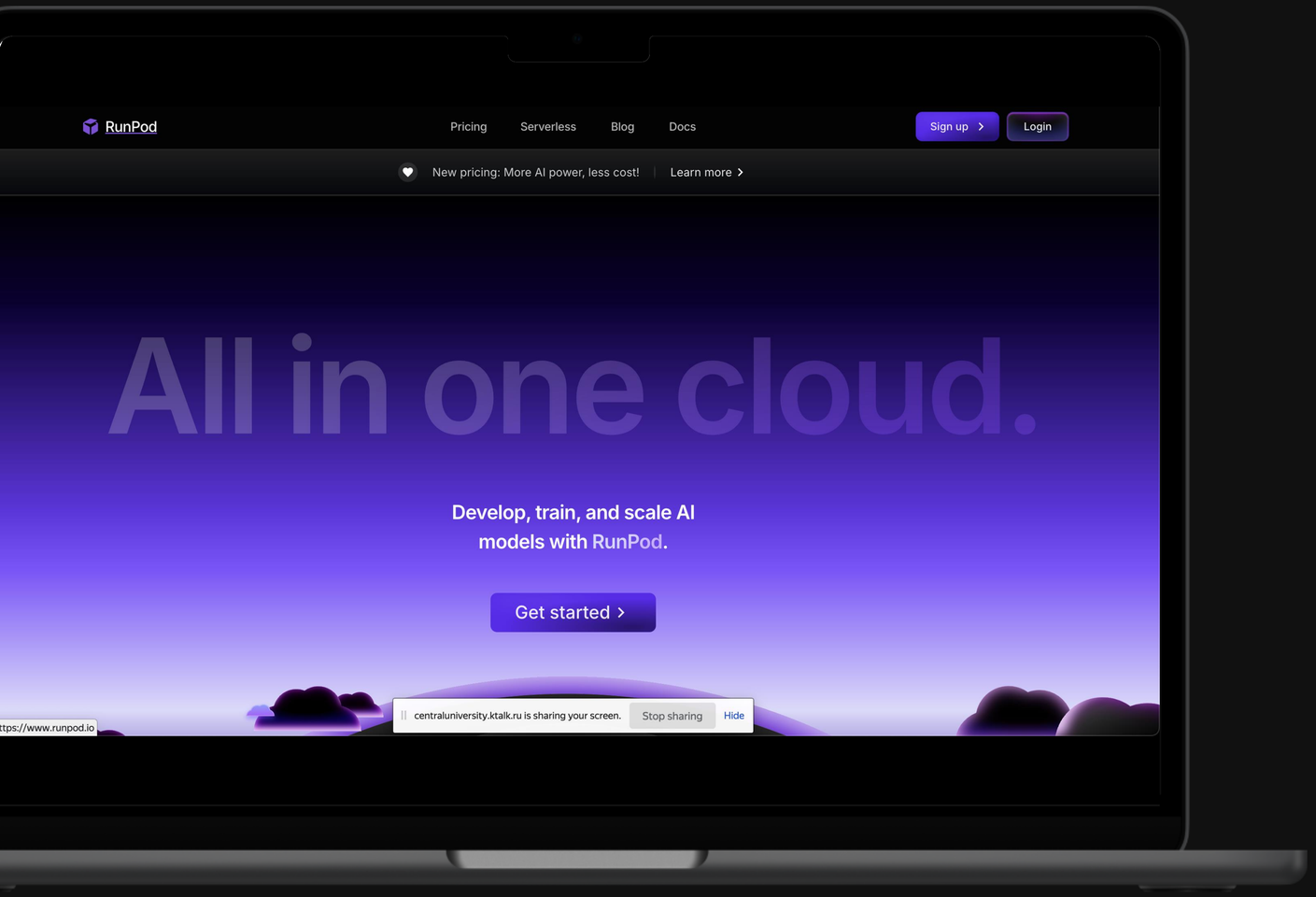
- 1 parameter = 4 bytes (32-bit float);
- 1B parameters = 4 x 10^9 bites = 4GB VRAM.

Useful calculators:

- LLM Tools;
- Can I run it? (LLM Version).



If you don’t even have a laptop: clo… services



Quick Deploy

Quick deploy custom endpoints with minimal config.

Text | Embedding | Image | Audio | All

Serverless vLLM

Deploy OpenAI-compatible Large Language Model (LLM) Endpoints efficiently with the vLLM Engine on Serverless in a few clicks.

Configure

Serverless SGLang

SGLang is fast serving framework for large language models and vision language models.

Configure

Quick Deploy

Quick deploy custom endpoints with minimal config.

Text | Embedding | Image | Audio | All

Infinity Vector Embed...

Deploy a wide range of text-embedding models and frameworks using Infinity on RunPod serverless.

Configure

https://www.runpod.io/console/serverless

Frameworks — orchestrators

LangChain

- Very popular..
- Good for rapid prototyping.
- Cool tools for document parsing

- Tons of dependency issues.
- Chaotic documentation and versioning
- Can miss critical bits
- Hard to customise.

LlamaIndex

- Less chaos, more structure — a neater, better-organised tool.

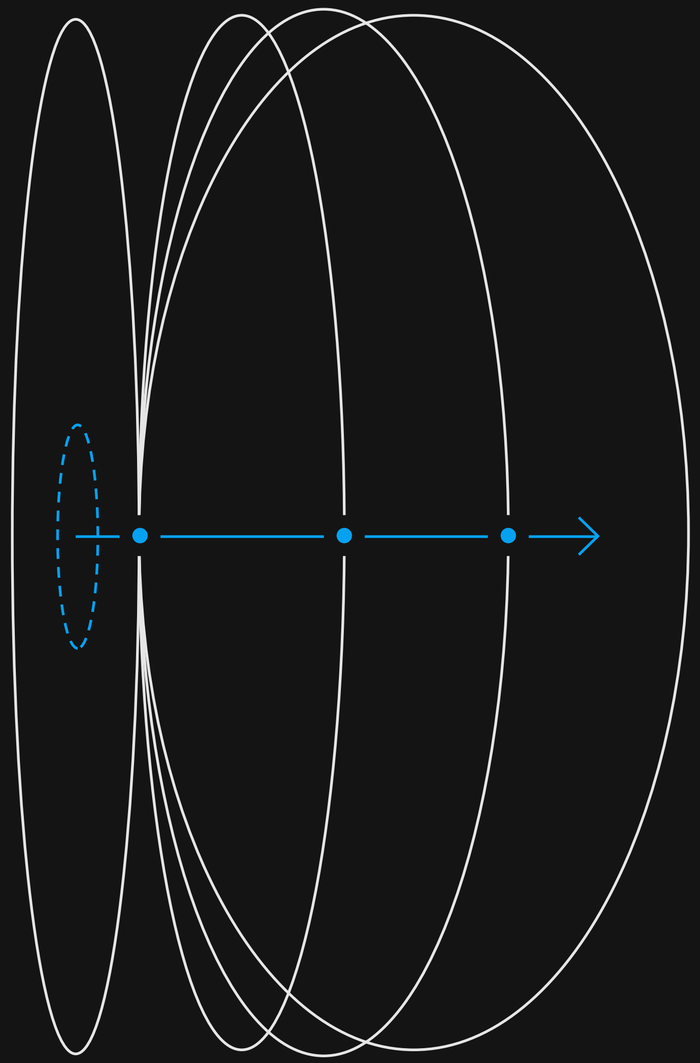
- Fewer integrations. Some parts are closed-source; for instance, the high-quality **LlamaParse** document parser is only available through a closed API.

HayStack

- Simple and easy to use.
- Often appears in job requirements

- Not as widespread; information can be hard to find

Conclusions



01 Choose between closed-source and open-source LLMs according to your task’s context.

02 If you can’t use closed APIs but you **have GPUs** — pick a SOTA framework (currently vLLM, though nuances already exist).

03 If you **lack GPUs or have few of them** — apply quantisation and look towards **llama.cpp**.

04 Study tuning flags carefully: default examples may be incomplete.

ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

Questions?